

Java EE 6: Develop Database Applications with JPA

Duration: 4 Days

What you will learn

This Java EE 6: Develop Database Applications with JPA NEW training explores using the Java Persistence API within the context of a web-based Java Enterprise Edition application, as well as within a stand-alone Java Standard Edition application. This includes using Java Persistence API with the Enterprise JavaBeans technology.

Learn To:

- Update multiple database tables based on relationships.
- Perform CRUD operations with JPA in Java SE and EE environments.
- Perform data validation using Bean Validation.
- Optimize JPA for performance.
- Apply transactions and locking.
- Map relational database tables to Java using ORM techniques and JPA.
- Understand key concepts found in the Java Persistence API.
- Create robust entity models.
- Create static and dynamic queries using Java Persistence API Query Language.
- Create type-safe queries with the Java Persistence API Criteria API.

Benefits to You

Learn how to accelerate the development of applications that use relational databases by mapping tables and table relationships to Java objects using Java Persistence API. You will also see how JPA solves issues with traditional relational database applications, including SQL injection.

JPA Enhancements

JPA has been enhanced and simplified in Java EE 6. The Java Persistence API (JPA) version 2.0 specification facilitates more effective and reliable (that is, more strongly typed) methodology for building object-centric criteria-based dynamic database queries.

JPA was introduced in Java EE 5, and provides a POJO-based persistence model for Java EE and Java SE applications.

Relational Data Mapping

Persistence is the technique through which object models broker the access and manipulation of information from a relational database. JPA handles the details of how relational data is mapped to Java objects, and it standardizes Object/Relational mapping.

Audience

Database Designers
J2EE Developer
Java Developer
Java EE Developer

Related Training

Required Prerequisites

Experience with Java EE 6 platform recommended

Experience with Java programming

Experience with Relational Databases recommended

Java SE 7 Programming

Developing Applications with Java EE 6 on WebLogic Server 12c

Suggested Prerequisites

Experience building and deploying EE applications

Experience with NetBeans recommended

Oracle WebLogic Server 12c Basic Administration Tasks OBEs

Course Objectives

Map relational database tables to Java using ORM techniques and JPA

Perform CRUD operations with JPA in Java SE and EE environments

Update multiple database tables based on relationships

Perform data validation using Bean Validation

Apply transactions and locking

Optimize JPA for performance

Course Topics

Course Introduction

Describing the target audience for this course

Explaining the course itinerary

Describing the format that the class will use

Introducing the course environment

Describing the need for Object-Relational Mapping

Introduction to Java Persistence API

Describing the Java Persistence API

Creating entity classes

Using persistent field and properties

Using a generated primary key (table, sequence and identity)

Obtaining an Entity Manager

Creating a Persistence Unit

Using an entity manager to create, find, update, and delete entities

Creating typed queries in JPA

Working with JPA in a Java Enterprise Environment

Evaluating the role of the container with JPA

Accessing JPA entities from a servlet

Evaluating the application of JSF as a user interface framework

Accessing JPA entities from Enterprise JavaBeans

Determining the impact of using stateless, stateful, and singleton session beans on entities

Configuring a persistence context in an EE context

Introduction to the Auction Application Case Study

Describing the auction application

Defining the domain objects of the auction application

Describing the implementation model for the auction system

Modeling Relational Databases with JPA Entities

Examining relationships in the data and object models

Using relationship properties to define associations

Implementing one-to-one unidirectional and bidirectional associations

Implementing many-to-one/one-to-many bidirectional associations

Implementing many-to-many unidirectional and bidirectional associations

Using OrderBy and OrderColumn annotations to define sort order

Applying the OrphanRemoval annotation to prevent orphaned entities

Working with the Entity Manager

Describing the relationship between an entity and an entity manager, and between a persistence context and a persistence

Differentiating between transaction-scoped and extended entity managers

Describing the entity life cycle

Using entity manager operations to perform CRUD operations: persist, find, merge, remove

Examining the role of the entity manager with detached entities

Defining and use cascading operations

Persisting Enums and Collections

Persisting entities that contain enums

Persisting entities that contain collections

Persisting entities that contain Maps

Creating Queries with the Java Persistence Query Language (JPQL)

Describing the Java Persistence Query Language (JPQL)

Contrasting JPQL with native queries

Using conditionals to filter results

Refining queries to return only needed data

Performing joins between entities

Creating dynamic queries with parameters

- Using named queries
- Performing bulk updates and deletes

Using the Criteria API

- Contrasting the Criteria API with JPQL
- Using the Criteria API structure and core interfaces
- Creating SELECT, FROM, and WHERE clauses
- Creating paths and expressions
- Using ORDER BY, GROUP BY, and HAVING clauses
- Using the canonical metamodel

Implementing Bean Validation with JPA

- Describing the JPA lifecycle phases where validation takes place
- Creating an entity listener class
- Utilizing validation groups
- Using built-in validation constraint annotations provided by Bean Validation
- Creating a custom Bean Validation constraint

Applying Locking and Transactions

- Describing transaction semantics
- Comparing programmatic and declarative transaction scoping
- Using JTA to scope transactions programmatically
- Implementing a container-managed transaction policy
- Supporting optimistic locking with the versioning of entity components
- Supporting pessimistic locking by using EntityManager APIs
- Describing the effect of exceptions on transaction state

Advanced Modeling: Entity Inheritance Relationships

- Evaluating object-relational mapping strategies for entity inheritance
- Applying single-table-per-class, joined-subclass, and table-per-class inheritance mapping strategies
- Using embeddable classes
- Overriding mappings with the @AttributeOverride and @AssociationOverride annotations
- Specifying composite primary keys

Optimizing JPA Performance

- Using lazy fetching to prevent the loading of entities that are not being used
- Using pagination to control the amount data that is needed at any one time
- Modifying queries to prevent the N + 1 problem
- Creating read-only queries
- Describing performance issues associated with IDENTITY ID generation
- Creating and using stored procedures with JPA and EclipseLink
- Using cache optimizations with JPA and EclipseLink